Date of publication xxxx 00, 0000, date of current version xxxx 00, 0000.

Digital Object Identifier 10.1109/ACCESS.2017.DOI

Matrix Factorization with Rating Completion: an Enhanced SVD Model for Collaborative Filtering Recommender Systems

XIN GUAN[1], CHANG-TSUN LI[1,2], AND YU GUAN.[3],
[1]Department of Computer Science, University of Warwick, UK (e-mail: x.guan,c-t.li@warwick.ac.uk)
[2]School of Computing and Mathematics, Charles Sturt University, Australia (e-mail: chli@csu.edu.au)
[3]School of Computing Science, Newcastle University, UK (e-mail: yu.guan@ncl.ac.uk)

Corresponding author: Chang-Tsun Li (e-mail: chli@csu.edu.au).

This work is supported by the EU Horizon 2020 - Marie Sklodowska-Curie Actions through the project entitled Computer Vision Enabled Multimedia Forensics and People Identification (Project No. 690907, Acronym: IDENTITY).

ABSTRACT Collaborative filtering algorithms such as matrix factorization techniques are gaining momentum recently due to their promising performance on recommender systems. However, most collaborative filtering algorithms suffer from data sparsity. Active learning algorithms are effective in reducing the sparsity problem for recommender systems by requesting users to give ratings to some items when they enter the systems. In this paper, a new matrix factorization model, called *Enhanced SVD* (*ESVD*) is proposed, which incorporates the classic matrix factorization algorithms with ratings completion inspired by active learning. In addition, the connection between the prediction accuracy and the density of matrix is built to further explore its potentials. We also propose the *Multi-layer ESVD* (MESVD), which learns the model iteratively to further improve the prediction accuracy. To handle the imbalanced datasets that contain far more users than items or more items than users, the *Item-wise ESVD* (IESVD) and *User-wise ESVD* (UESVD) are presented, respectively. The proposed methods are evaluated on the famous *Netflix* and *Movielens* datasets. Experimental results validate their effectiveness in terms of both accuracy and efficiency when compared with traditional matrix factorization methods and active learning methods.

INDEX TERMS matrix factorization, recommender systems, data sparseness, rating completion, active learning

I. INTRODUCTION

RECOMMENDER systems are one of the most common software tools and techniques for generating recommendations since the early 1990s. They provide users with personalized recommendations by predicting the preference (often expressed in rating) that the users would give to an item, and typically apply methodologies and techniques from related areas such as *Machine Learning*, *Information Retrivial*, and *Human Computer Interaction*. They play vital roles in real life and are adopted by many internet leaders such as *Google* [1], *Facebook* [2], *Amazon* [3], *Netflix* [4], etc.

Recommender systems are used for generating recommendations to users, usually in one of the following ways:

- Collaborative filtering algorithms [5] predict other items the current users might like based on the past knowledge about the preferences of users for some items.
- Content-based algorithms [6] produce recommendations based on item descriptions which could be automatically extracted or manually created, or (and) user profiles that represent the users' interests on items.
- Knowledge-based algorithms [7] generate recommendations by exploiting explicit user requirements and detailed domain knowledge about item features, reasoning about what items meet the user's needs.
- Hybrid approaches [8] generate recommendations by combining several algorithms or recommendation components, which are based on the above three approaches: collaborative filtering and content-based and knowledge-based algorithms.

Collaborative filtering is considered the most important techniques, and is widely used in industry, especially in online retail sites such as *Netflix* [4], in order to promote additional items and increase sales. It is a method that makes recommendations by using ratings given to items by users as the only source of information. Empirical studies such as [9] and [10] categorized collaborative filtering algorithms into two classes: memory-based and model-based algorithms. Memory-based algorithms [11] [12] focus on relationships between users (user-based) or items (item-based), while model-based approaches [13] [14] [15] are based on prediction models that have been trained using the rating matrix. Matrix factorization methods [16] [17] [18], as one of the most successful realizations of model-based algorithms, can achieve better accuracy than classic nearest neighbor methods (memory-based) when dealing with product recommendations [19].

In real-life scenarios, when a new user comes in, most recommender systems would only query the user to rate a limited number of items (which is a small proportion comparing with the whole data set). Therefore, there is not enough knowledge to form accurate recommendations for the user since the rating matrices are normally very sparse. To get precise recommendations for the target user, active learning is often used to elicit more high-quality data [20] [21] [22].

However, traditional active learning methods [23] [24] [25] only evaluate each user independently and only consider the benefits of the elicitations to the "new" user, neglecting the effects of the elicitations to the whole system. Moreover, in previous works [9] [23] [24], selected users are enforced to rate each elicitation through an active learning process, which is unrealistic in practice. In this paper an *ESVD* model is proposed by incorporating the *matrix completion* strategy, which improves the prediction accuracy of the whole system by automatically 'adding' ratings only for existing users. Furthermore, ratings were added on a one by one per request [23] or user's by user's per request basis [25] by the system in traditional active learning. As a result, the model is trained at each request, which is considerably time-consuming. In contrast, the proposed *ESVD* model increases the amount of special training ratings simultaneously that are predicted by matrix factorization. Through this special preprocessing step, not only is the computational cost reduced, but also the performance of the recommender system is greatly improved. To alleviate the bias of the added ratings in the *ESVD* model, we also propose the *Multilayer ESVD* (*MESVD*) algorithm, which learns the model iteratively. To handle the imbalanced datasets that contain far more users than items or more items than users, we also propose the *Item-wise ESVD (IESVD)* and *User-wise ESVD (UESVD)*, respectively. The connections of the proposed *ESVD* and its variants are shown in Figure 1.

The rest of the paper is organized as follows: Preliminaries and related works are introduced in Section II. In Section III we first analyze active learning, then propose the *ESVD* model for recommender systems. Section IV describes the *MESVD* model to further explore its potential. More extensions on *ESVD* including *IESVD*, *UESVD* are presented in Section V to handle the imbalanced datasets. The paper is concluded in Section VI.

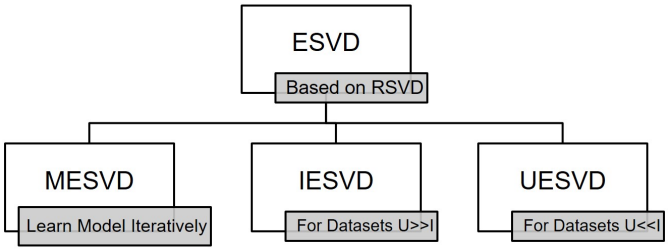


FIGURE 1: The proposed *ESVD* and its variants

II. PRELIMINARIES

Collaborative filtering recommender systems usually consist of a set of users, a set of items and the preferences of users for various items, which are frequently represented in the form of [User, Item, Rating] triples. By aggregating these triples, a U-I rating matrix $R \in \mathbb{R}^{m \times n}$ that consists of m users and n items can be obtained, where each rating r_{ij} represents the preference of user i for item j. The task of collaborative filtering recommender systems is to recommend each user a list of unrated items that are ranked in a descending order based on predicted preferences (ratings). As the key point of collaborative filtering is the ratings prediction task, most algorithms transform the recommending problem into the missing value estimation problem in the U-I rating matrix with high sparsity. The evaluation of the algorithms is often measured by computing the prediction accuracy of a set of unknown ratings in the rating matrix based on the predefined metrics, such as *Root Mean Squared Error (RMSE)* and *Mean Absolute Error (MAE)*.

As the knowledge of preferences is very limited, the rating matrices in most recommender systems are extremely sparse. Take *Netflix* [4] and *Movielens* [26] datasets as examples, the density are 1.18% and 6.3%, respectively, which mean that only a few elements are rated. Another challenge is that the dataset used in real-world recommender systems is typically of high dimensionality. Due to high sparseness and computational complexity, directly applying traditional dimensionality reduction methods, like *SVD* algorithms, to rating matrices is not appropriate [27].

A. REGULARIZED SVD

Funk [28] proposed the *Regularized SVD* (*RSVD*) algorithm, which is proved to be effective for collaborative filtering. Denote the rating matrix by $R \in \mathbb{R}^{m \times n}$ that consists of m users and n items, and the prediction of the rating matrix by $\tilde{R}$. The

Regularized SVD algorithm decomposes the rating matrix R into the products of two lower rank matrices $U \in \mathbb{R}^{k\times m}$ and $V \in \mathbb{R}^{k\times n}$ as the feature matrix of users and items:

$$\tilde{R} = U^T V \tag{1}$$

It is based on an assumption that each user's rating is composed of the sum of preferences about various latent factors of that item. So each rating r_{ij} (the corresponding prediction is represented as $\tilde{r}_{ij}$) the ith user gives to the jth movie can be represented as:

$$\tilde{r}_{ij} = {U_i}^T V_j \tag{2}$$

where U_i, V_j are the feature vectors of the ith user and the jth movie, respectively.

To learn the optimized approximations of U and V, the system minimizes the sum of squared errors between the existing scores and prediction values:

$$E = \frac{1}{2}\sum_{i,j\in\kappa}(r_{ij} - \tilde{r}_{ij})^2 + \frac{k_u}{2}\sum_{i=1}^{m}{U_i}^2 + \frac{k_v}{2}\sum_{j=1}^{n}{V_j}^2 \tag{3}$$

where κ is a set of *(u,i)* pairs that has been assigned values originally in the rating matrix R (a.k.a. the training set), the regularization parameters k_u and k_v are used to alleviate over-fitting.

To solve the optimization problem like Equation (3), *Stochastic Gradient Descent (SGD)* is widely used and has shown to be effective for learning matrix factorization models [16] [29]. It loops through training ratings (κ) and modifies U and V in the opposite direction of the gradient:

$$U_i \leftarrow U_i - \alpha\frac{\partial E_{ij}}{\partial U_i} \tag{4}$$

$$V_j \leftarrow V_j - \alpha\frac{\partial E_{ij}}{\partial V_j} \tag{5}$$

where α is the learning rate.

Unlike traditional *SVD*, *RSVD* is a procedure of optimizing the feature matrices, which minimizes the least square error of the approximations. By solving this optimization problem, the end result is the same as *SVD*, which gets the diagonal matrix arbitrarily rolled into the two side matrices.

B. DATASETS

Movielens [26] is a classic recommender system that recommends movies for its users through collaborative filtering algorithms. *Movielens 1M* contains 1,000,209 anonymous ratings of 3,952 movies provided by 6,040 users, while *Movielens 100K* contains 100,000 ratings from 943 users on 1,682 movies.

Netflix [4] is a recommender system consisting of over 100 million 5-star ratings by 480,189 users for 17,770 movies. Each rating in the *Movielens* and *Netflix* datasets is an integer ranging from 1 to 5, which represents the level of perferences the user has for the corresponding movie. The first 106,150 ratings are extracted from the full *Netflix* dataset as the subset of *Netflix*, which are provided by 1,910 users for 1,780 movies.

III. THE PROPOSED ENHANCED SVD (ESVD) MODEL

It is important to note that the characteristics of prediction algorithms may influence the prediction accuracy. Matrix factorization models like *Regularized SVD (RSVD)* are learnt by fitting a limited number of existing ratings. As a result, the model trained with good quality as well as large quantity ratings could achieve better performance than the one with less sufficient ratings. Generally, the more number and informative ratings are obtained, the better performance recommender systems could achieve. However, most recommender systems suffer from the sparsity problem, i.e. the rating matrices are extremely sparse since users are often unwilling to rate a large amount of items.

A. CLASSIC ACTIVE LEARNING ALGORITHMS

Active learning is widely used for obtaining high quality data that better represents the preferences of users. To achieve this purpose, the system requests the user to rate specific items based on certain strategies or criteria. These strategies include:

1) *Randomization*: The system selects items or users to present randomly with uniform probability over all the items or users, which can be regarded as the baseline strategy for comparison [30].
2) *Popularity-based*: Items or users with the highest number of ratings are preferred, which is based on the assumption that the user is more likely to give ratings to popular items [30].
3) *Variance-based*: The system selects items with the largest variance for eliciting. Therefore, the items with the largest variance are preferred for reducing the certainty of the system [30].
4) *Similarity-based*: The system selects the items with the highest similarity to the user's previously rated items.
5) *Hybrid*: This includes $Sqrt(Frequency) * Variance$ [31], *Voting* [25] , which consider the overall effect of previous methods.

These strategies try to identify the most informative set of training examples, aiming to achieve better performance for users with a certain amount of ratings required from them. However, tradition active learning has several limitations:

1) First, previous works (e.g., [25] [30] [32]) focused on the accuracy of the recommendations for 'a single user', regardless of the fact that the increase of elicitations affect the performance of the whole system.
2) Furthermore, the model was trained by iterating all the users, which incurs high computational costs. With classic active learning strategies, the items selected for different users to elicit are not always the same. For example, the items with the highest similarity for a user's previously rated items may not be the same

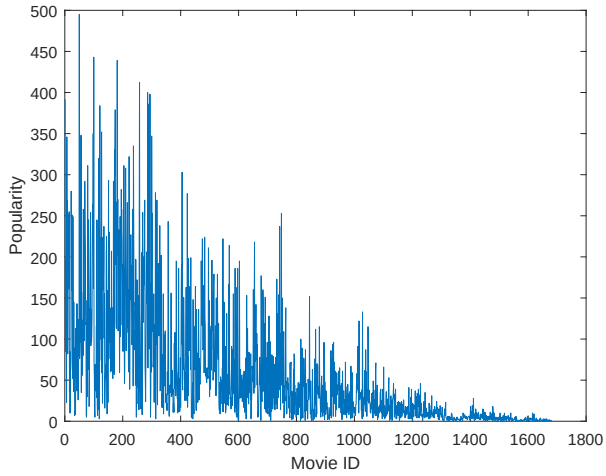


FIGURE 2: The number of ratings each item has received (popularity) in *Movielens 100K*

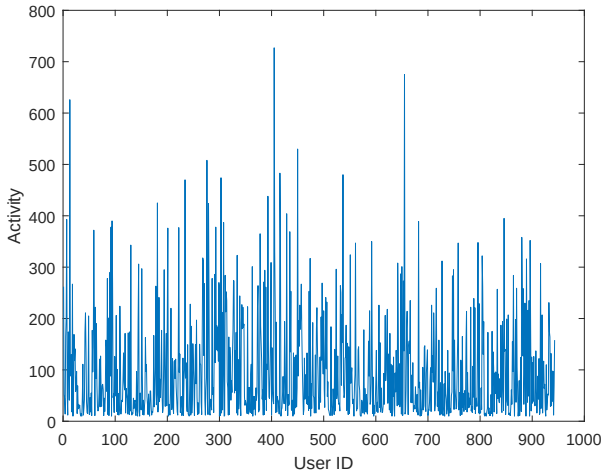


FIGURE 3: The number of ratings each user has rated (activity) in *Movielens 100K*

as another user's since different users have different experiences. Hence the strategy (personalized) has to be applied repeatedly for each user, in order to elicit different items.

3) In addition, current active learning methods are based on the assumption that a user can provide ratings for any queried items, which is unrealistic and costly. Take movie recommendation for example, to rate a movie that is generated by the active learning strategy, a user has to watch it. On the other hand, the user maybe be frustrated when asked to rate a movie that he/she has not watched.

B. THE PROPOSED METHODS

1) Seeking High Density through Popular Items

Usually the number of ratings to each item (popularity) varies significantly. Take the *Movielens 100K* dataset for example, the minimum and maximal number of popularity is 0 and 495, respectively, as shown in Figure 2. Popularity is based on the number of ratings for each item only. Therefore, the popularity of each movie remains the same for all the users. Based on the assumption that users are more likely to rate world-famous movies than the less known movies, by selecting N most popular movies for all the users, a new sub-matrix could be obtained. Then the unrated items in this sub-matrix would be the 'desirable' movies in some sense for the users who missed before. Unlike traditional active learning that queries only new users for a certain number of ratings in each iteration, the proposed strategy is to predict these specific ratings for all the users at the same time (one iteration) based on the result of applying matrix factorization algorithms to this sub-matrix. After these ratings are added to the original rating matrix, a more accurate matrix factorization model could be trained.

In summary, this item-oriented (based on item popularity) approach pre-estimates ratings of only popular movies for all the users simultaneously (instead of iterating through each individual user) in order to improve the performance of the whole system. Therefore, it reduces the training iteration from as high as the number of users (for active learning) to only 2 (the proposed method), which significantly reduces the computational cost.

2) Seeking High Density through Active Users

In contrast to traditional active learning for collaborative filtering, which selects a number of items to rate so as to improve the rating prediction for the user, Carenini et al. [33] proposed an alternative active learning method that elicits ratings by querying a number of special users about a specific item in order to improve the rating prediction for the target item. Inspired by [33], we also propose a user-oriented approach to further explore the potential of the proposed method.

Generally, the number of movies each user has rated (activity) varies significantly as shown in Figure 3. Take movie recommendations as an example, though active users who are enthusiastic about movies may watch far more than the ones who are not into movies, there are likely to be movies that the users have watched but not yet rated. Therefore, it is easier to accept that active users are more likely to give ratings to their unrated movies. These movie enthusiasts are selected based on the number of rated movies. Ratings of the unrated items (as the missing values in the new sub-matrix) would be predicted by matrix factorization algorithms. These new ratings are then added to the original matrix for generating better recommendations.

In brief, this user-oriented approach (based on user activity) tries to improve the performance of the whole system by adding pre-estimated ratings simultaneously of all items provided by active users only. Therefore, it also has the benefits that item-oriented approach has. However, both algorithms may still incur significant computational cost and distortion

of the original model because of the extensive selection of added ratings, especially when a large number of popular items or active users are selected in the sub-matrix.

3) Proposed *ESVD* - Density-Oriented Approach

So far an item-oriented approach and a user-oriented approach are presented, both based on the idea that pre-estimating a set of selected ratings simultaneously for the matrix factorization model to learn. The rationale behind the pre-estimates is that they are predicted from a denser sub-matrix, which allows more accurate elicitations of the missing ratings. Typically the denser the matrix is, the better the matrix factorization model is obtained. For example with the *Movielens 100K* dataset, the sub-matrix with 5% of the most popular movies is of density 29.47% in contrast to the orginal matrix (6.3%). On the other hand, the density of the new sub-matrix by choosing 5% of the most active users is 23.33%, which still much denser than the original one. Based on this observation, we propose a density-oriented approach, called *ESVD* as shown in Algorithm 1 and Figure 4), which combines the afore-mentioned item-oriented and user-oriented high density seeking strategies.

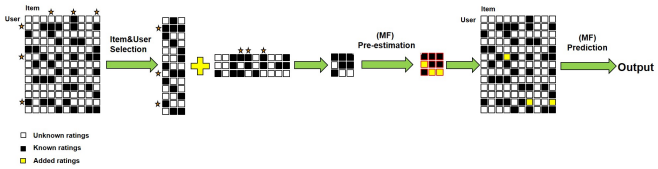


FIGURE 4: Procedures of *ESVD*

ESVD is based on the assumption that the recommender system was first built with a set of the most popular movies that are rated by a set of the most active users. Because item popularity depends on the number of ratings each user provides and user activity is related to the number of ratings each item receives, by choosing the N most popular items (columns) and the N' most active users (rows), a much denser sub-matrix is obtained (as shown in Figure 4). Take the *Movielens 100K* dataset as an example, the newly-formed sub-matrix would reach to 77.28% density by selecting 5% of the most popular movies and 5% of the most active users (Step 3 in Algorithm 1). The missing values in this sub-matrix can be explained as ratings of the most famous movies but have not been rated by a group of the most active users. Therefore, the recommendations generated by this recommender system should be of high accuracy. Afterwards some rare movies most people probably have not seen and users who provide very few ratings are added to the dataset (the orginal matrix), which could lower the prediction accuracy of the whole system. To achieve better performance, the ratings generated from the sub-recommender system (pre-estimates) could be used (by applying matrix factorization on the sub-matrix) as the known knowledge for further learning and inference. Finally an enhanced matrix factorization model can be obtained with higher accuracy by learning the existing and extra ratings.

Algorithm 1 The Proposed *ESVD* (Density-Oriented Approach)

Input: Rating matrix $R \in \mathbb{R}^{m \times n}$, where $P_{i \in [1,m]} \in \mathbb{R}^{1 \times n}$ is the row vector and $Q_{j \in [1,n]} \in \mathbb{R}^{m \times 1}$ is the column vector;
T, test set in rating matrix *R*;
N, the number of items selected in the sub-matrix based on popularity;
N', the number of users selected in the sub-matrix based on activity ;

Output: *RMSE* of the test set T;

Step 1: Sort both items $j(1), j(2), ..., j(m)$ and users $i(1), i(2), ..., i(n)$ in the descending order based on popularity and activity, respectively;

Step 2: Create a sub-matrix M_1 by selecting the top N items (columns) of R based on the popularity such that $M_1 = [Q_{j(1)}, Q_{j(2)}, ..., Q_{j(N)}](N < m)$;
Creat a sub-matrix M_2 by selecting the top N' users (rows) of R based on the activity such that $M_2 = [P_{i(1)}, P_{i(2)}, ..., P_{i(N')}](N' < n)$;

Step 3: Create a sub-matrix M_3 by selecting the intersection of the top N items (columns) and top N' users (rows) based on the popularity and activity. Therefore, $M_3 = M_1 \bigcap M_2$;

Step 4: Apply basic matrix factorization (*Regularized SVD*) on matrix M_3 to obtain feature matrices U and V according to Equation (1). Then predict every missing value in sub-matrix M_3 to acquire a non-null matrix M_3' according to Equation (2). Then form a series of ratings L such that $L = \{r_{i_{k(1)}, j_{t(1)}}, ..., r_{i_{k(n)}, j_{t(n')}}\}$, where $r_{i_k, j_t} \in (M_3 \bigcap \neg\kappa)$;

Step 5: Fill the missing ratings in the original matrix R with the value predicted in **Step 4** to acquire a new rating matrix R'. That means the extra ratings are added into the training set $\kappa = \{\kappa, L\}$;

Step 6: Apply basic matrix factorization (*Regularized SVD*) on matrix R' to obtain feature matrices U' and V' according to Equation (1). Then predict the target ratings (test set) according to Equation (2) and calculate *RMSE* of the test set *T* according to Euqation (6);

C. EVALUATION

1) Experimental Setup

Experiments of the proposed algorithms (*ESVD-I*, *ESVD-U* and *ESVD*) are conducted on the classic recommender system datasets: *Movielens 100K* and the subset of *Netflix*. Some experiments with the larger version are also performed and similar results are obtained. However, it requires much longer time to perform the experiments since the models are trained and tested each time for different choice of N and N'. Therefore, the smaller datasets *Movielens 100K* and the subset of the original *Netflix* are used in order to run

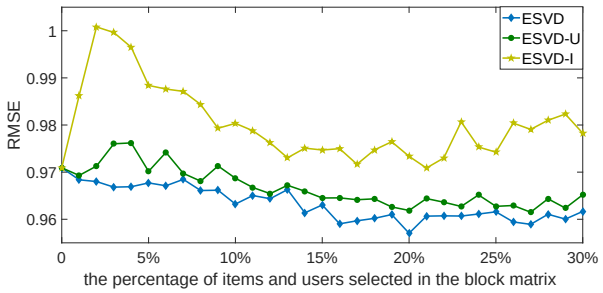


FIGURE 5: *Movielens 100K*: *RMSE* Comparisons of the proposed methods based on *RSVD*. Note the results corresponding 0 percentage in the plot is the results the the baseline RSVD.

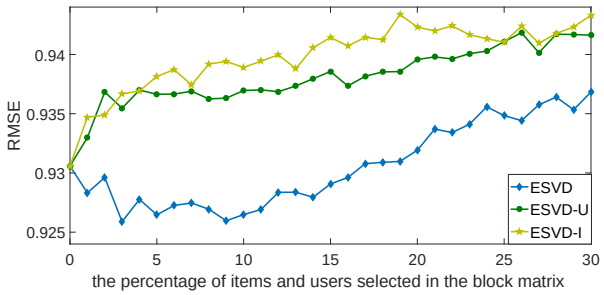


FIGURE 6: *Netflix*: *RMSE* Comparisons of the proposed methods based on *RSVD*. Note the results corresponding 0 percentage in the plot is the results the the baseline RSVD.

more experiments so as to evaluate how these two parameters affect the results of the proposed algorithm.

Normally each dataset is partitioned into a training set and a test set. The model is trained on the training set and the quality of results is usually measured by the *Root Mean Square Error* (*RMSE*) of the test set:

$$RMSE = \sqrt{\frac{\sum_{(\{i,j\} \in TestSet)} (r_{ij} - \tilde{r}_{ij})^2}{T}} \tag{6}$$

where T represents the total number of *(u,i)* samples in the test set. *RMSE* is used as the default metric which is widely used in the *Netfilx* Competition [4] and proved to be effective for measuring recommender systems.

The number of the latent factors (rank) k is a key factor for matrix factorization models. Increasing k would result in better prediction at a higher computational cost. Here we set $k = 10$ for training each matrix factorization model in both datasets that balances the performance and training time. To learn the matrix factorization model from the sub-matrix, the regularization coefficient k_u and k_v in Equation (3) are set to 0.01 and 0.05 for the *Movielens 100K* and *Netflix* datasets, respectively. The learning rate α in Equation (4) is set to 0.1 and is decreased by a factor of 0.9 each iteration for both datasets. To learn the matrix factorization model in the rating matrix R' (with pre-estimates), the regularization parameters $k_u{}'$ and $k_v{}'$ are set to 0.1 for both datasets. The learning rates α are set to 0.01 and 0.05 initially and allowed to decrease by a factor of 0.9 each iteration for the *Movielens 100K* and *Netflix* datasets, respectively.

2) Experimental Results

We use *ESVD-I*, *ESVD-U* and *ESVD* to identify our methods proposed in Section III.B.1, III.B.2, and III.B.3, respectively. Figure 5 and Figure 6 show the results of the proposed methods based on the number of items and users selected (simply setting $N = N'$ in this case) in the sub-matrix on the *Movielens 100K* and *Netflix* datasets, respectively. All the methods start at 0 point where no extra filling is added into the learning process, which is the same as *RSVD*. It can be seen that the results of *ESVD-I* and *ESVD-U* sometimes are not promising. This is because the pre-estimates added to the training sets are either based only on item popularity (or user activity), which may lead bias and distort the latent factor model. For example, most people prefer movies with a happy ending, and hence popular movies always contain more comedies than tragedies. As a result, more ratings of comedy movies would be added to the learning process by the users, which leads to more weights added to the factor corresponding to comedies in the latent factor model (*RSVD* in this case) and deteriorate the performance. It is apparent from Figure 5 and Figure 6 that the proposed *ESVD* consistently outperforms other methods including the baseline method, *RSVD*.

TABLE 1: *RMSE* of *ESVD* on *Movielens 100K* (The Density-Oriented Approach)

Items&Users	Sub-matrix Density	Extra Ratings	*RMSE*
0%	null	null	0.9709
5%	77.28%	897	0.9677
10%	65.20%	5496	0.9632
15%	53.90%	16381	0.9630
20%	45.66%	34508	0.9570

TABLE 2: *RMSE* of *ESVD* on *Netflix* (The Density-Oriented Approach)

Items&Users	Sub-matrix Density	Extra Ratings	RMSE
0%	null	null	0.9306
5%	59.06%	3498	0.9265
10%	43.59%	19179	0.9265
15%	33.10%	51268	0.9291
20%	25.51%	101298	0.9319

Table 1 and Table 2 illustrate the experimental results of the proposed density-oriented (*ESVD*) approach on the *Movielens 100K* and *Netflix* datasets. The results of the performance (in terms of *RMSE*) are shown based on the percentages of items and users ($N = N'$ from 0% to 20%) selected. Note that the basic *RSVD* is a special case of the proposed approach with $N = 0\%$, which is used as the baseline for comparision. It can be observed that the density of the sub-matrix decreases with more items and users selected from the original matrix since the items and users are chosen based on the number of ratings. Although

sparser matrices may result in poor quality of pre-estimates, the quantity is increased as the size of the sub-matrix gets large. Therefore, more missing values can be pre-estimated by matrix factorization on the sub-matrix and put into the learning process of the original model. Because the sub-matrix of the *ESVD* approach is the intersection of the N items (item-oriented) and N' users (user-oriented) with the largest number of ratings, its density is much greater than the ones used for *ESVD-I* and *ESVD-U*. Even with fewer ratings to be added compared with *ESVD-I* and *ESVD-U*, the performance is better.

In the experiments, it can be observed that for the *Movielens 100K* dataset, the performance of *ESVD* fluctuates as the number of users and items increases (Figure 5). While for the *Netflix* dataset (Figure 6), the *RMSE* of *ESVD* drops at first and then deteriorates (the lower *RMSE* the better performance) as *N* goes up. This is mainly because the *Netflix* dataset is much sparser than the *Movielens 100K* dataset. As *N* increases, more poor quality data is added into the learning process and leads to poorer performance (Figure 6). The optimal value for *N* that balances the quality (density of sub-matrix) and the quantity (number of added ratings) depends on the distribution of ratings. For the *Movielens 100K* dataset, *ESVD* can reach 0.9570 (when $N = 20\%$) which reduces the *RMSE* by 0.0139 when compared with the *RSVD* 0.9709. For the *Netflix* dataset, *ESVD* can reduce the *RMSE* by 0.0047 from 0.9306 to 0.9259 when $N = 3\%$.

IV. PROPOSED MULTILAYER ESVD (MESVD)

With *ESVD*, all the extra ratings are predicted in a single matrix factorization model simultaneously, which could lead to significant bias and distort the original model when the number of pre-estimates is large. To alleviate this problem a method called *Multilayer ESVD* (*MESVD*) is proposed, which obtains the fillings incrementally through multiple matrix factorization on different sub-matrices.

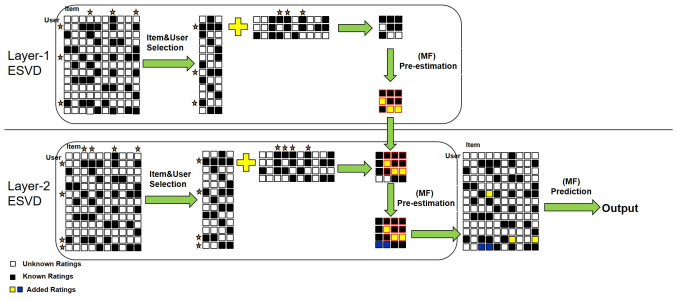


FIGURE 7: Procedures of *Multilayer ESVD*

The example of a *Two-layer ESVD* is shown in Figure 7. First a set of sub-matrices are created in each layer by selecting the intersections of different numbers of columns and rows (as starred) based on the number of ratings each item received or each user provided, respectively. Therefore, each smaller sub-matrix (with red frame) in the upper layer can be seen as a part of the bigger sub-matrix in the lower layer. The missing values (yellow ratings) in the smaller sub-matrix can be predicted by the matrix factorization method and then be regarded as known ratings in the bigger sub-matrix. Similar to deep learning, the outputs generated by each layer are utilized as the inputs of the next layer, for enhancing the prediction accuracy of their outputs (pre-estimates). For example in Figure 7, the ratings in black and yellow are known in layer-2. Therefore, the sub-matrix in the next layer is much denser than the one without pre-estimates (the ratings in yellow) from the first layer. In this way fillings are predicted iteratively layer by layer. At last all the pre-estimated ratings are added to the original matrix to evaluate the performance of the whole system.

Basically, the *MESVD* approach is based on the assumption that the recommender system is built by a very dense matrix with sufficient ratings first. Therefore, the recommendations (the missing values in the sub-matrix) were reliable and can be regarded as the known knowledge. After that, it is better to keep inviting the most active users to rate the most popular movies for the recommender system. In this way, each time a set of movies and users are added to the system, which generates knowledge iteratively for further learning and inference (from the upper layer to lower layer). As a result, better performance can be obtained by learning the current systems with extra knowledge generated in each of the sub-systems.

A. EXPERIMENTAL RESULTS

Experiments of the *MESVD* method on the *Movielens 100K* and *Netflix* datasets are also conducted. The corresponding results are shown in Table 3 and Table 4. For the *Movielens 100K* dataset, experiments of *ESVD* are conducted when $N = 20\%$ (optimal point), *Two-layers ESVD* where the first layer is 10% and the 2nd layer is 20%, *Four-layers ESVD* with layers from 5% to 20% with a 5% interval (setting $N = N'$). Specifically, in the first experiment, ratings are predicted from the sub-matrix of density 45.66%. In *Two-layers ESVD*, the first 5496 ratings are predicted from the sub-matrix of density 65.20%, while the rest are calculated from the sub-matrix of density 54.32%. In *Four-layers ESVD*, ratings are predicted successively, from where each time ratings are added based on a much denser matrix. Since we add the same amount of pre-estimates totally (34508) for each experiment, the sub-matrices in the final layers are the same correspondingly. However, the performance gets better from single layer *ESVD* to Four-layers *ESVD*, for the reason that the quality of pre-estimates gets better.

For the *Netflix* dataset, four experiments are performed: *ESVD* when $N = 10\%$, *Two-layers ESVD* where the first layer is 5% and the 2nd layer is 10%, *Four-layers ESVD* with layers from 2.5% to 10% with a 2.5% interval, and *Six-layers ESVD* with layers from 5% to 10% with a 1% interval. It can be observed that *Two-layers ESVD* yields better performance than *ESVD*, because each batch of extra ratings are predicted from the denser matrices with better accuracy. For the same reason, better results can be obtained based on *Four-layers*

TABLE 3: *RMSE* of *MESVD* on *Movielens 100K*

	Item&User	Sub-matrix Density	Extra Ratings	*RMSE*
RSVD	N=0	0	0	0.9709
ESVD	N=20%	45.66%	34508	0.9570
Two-layers ESVD	N=[10%, 20%]	65.20% 54.32%	5496 29012	0.9564
Four-layers ESVD	*N*=[5%, 10%, 15%, 20%]	77.28% 70.88% 69.37% 71.46%	897 4599 10885 18127	0.9561

TABLE 4: *RMSE* of *MESVD* on *Netflix*

	Item&User	Sub-matrix Density	Extra Ratings	*RMSE*
RSVD	N=0	0	0	0.9306
ESVD	N=10%	43.59%	19179	0.9265
Two-layers ESVD	N=[5%, 10%]	59.06% 53.88%	3498 15681	0.9262
Four-layers ESVD	N=[2.5%, 5%, 7.5%, 10%]	67.04% 67.39% 68.33% 71.72%	712 2786 6068 9613	0.9248
Six-layers ESVD	*N*=[5%, 6%, 7%, 8%, 9%, 10%]	59.06% 83.76% 84.35% 86.11% 86.74% 87.07%	3498 1998 2621 3017 3650 4395	0.9255

ESVD than *Two-layers ESVD*. When compared *Six-layers ESVD* with *Two-layers ESVD*, the first batch of extra ratings are the same. However, the rest are of better quality because they are learnt layer by layer in the denser matrices. When compared *Six-layers ESVD* with *Four-layers ESVD*, although all the pre-estimates are learnt through more iterations, the first batch of extra ratings are of poorer quality. As a result, the performance of *Six-layers ESVD* is not as good as *Four-layers ESVD*. In summary, although the optimal value for N is not selected, better performance is obtained through *MESVD* than *ESVD*.

Experimental results show that the quality of *MESVD* depends on the number of layers and the choice of each layer. With the *ESVD* algorithm, decent result cannot be obtained if the number of items and users selected in the sub-matrix *N* is inappropriate. Through the *MESVD* method, this problem can be alleviated with comparable or better results. With optimal value for *N*, better performance can still be obtained by learning the extra ratings iteratively through the *MESVD* method. The improvements of the *MESVD* approach is limited, as the ratings added in the original matrix are the same when compared with *ESVD* approach. However, if the training time is not a concern, *MESVD* is preferable.

V. PROPOSED EXTENSIONS OF ESVD

Experimental results presented in the previous sections show that adding pre-estimates by applying *ESVD* do improve the performance of the system. The reason behind is that the model is learnt by extra high quality ratings that are predicted from the dense sub-matrix based on item popularity and user activity. Based on this finding, we propose two extensions of *ESVD* to deal with the datasets with a substantial imbalance between the number of users and the number of items.

A. PROPOSED ITEM-WISE ESVD (IESVD)

When dealing with the rating matrix of which the number of users is far greater than the number of items, most items are rated by a significant number of users (popularity) but each user only rates a few items (activity) on average. Therefore, popular items have more impacts than active users on the density of the newly formed sub-matrix. As a result, obtaining a sub-matrix based on item popularity and user activity simultaneously is not appropriate under such circumstance.

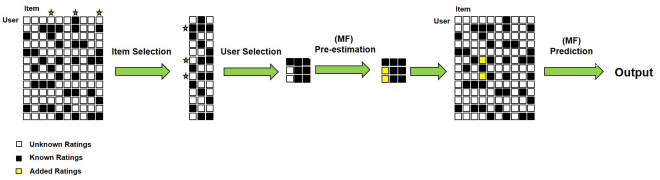


FIGURE 8: Procedures of *Item-wise ESVD*

The *Item-wise ESVD (IESVD)* is proposed by first selecting a number of most popular items to form a sub-matrix as the *ESVD* Algorithm does (Fig 8). Then, only the active users that have seen these specific movies (as starred) are chosen. In this way, a denser sub-matrix can be obtained than the one from the *ESVD* method. Therefore, the missing values in the sub-matrix can be pre-estimated by the matrix factorization method. Finally, the predicted ratings are filled in the original matrix. Therefore, the new matrix factorization model is learnt and tested based on the newly formed rating matrix.

B. PROPOSED USER-WISE ESVD (UESVD)

Likewise, in the datasets consisting of far more items than users, the quantity of ratings each user rates (activity) is much greater than the quantity of ratings each item is rated (popularity) on average. Therefore, the *User-wise ESVD* (*UESVD*) is proposed as shown in Fig 9. Initially, a number of most active users are selected to form a sub-matrix based on the number of ratings each user has rated. Then the most popular items that the active users have seen are chosen to form the sub-matrix, i.e. the items with most ratings in the sub-matrix only. As a result, a denser sub-matrix is obtained than the one from *ESVD*. The rest procedures are the same as the *ESVD* algorithm.

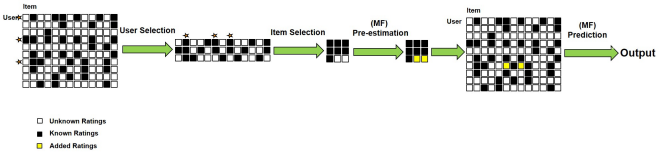


FIGURE 9: Procedures of *User-wise ESVD*

Therefore, both *IESVD* and *UESVD* train the matrix factorization model twice by automatically adding pre-estimates in the dataset. However, the *IESVD* and *UESVD* approaches are not applicable to multilayer learning (i.e. *MESVD*) because in

the *IESVD* and *UESVD* algorithms, the sub-matrices selected based on few items and users are not necessarily included in the larger sub-matrices consisting of more items and users.

C. EXPERIMENTAL RESULTS

To emphasize the merits of the proposed *IESVD* and *UESVD* approaches, the following two subsets are extracted from *Movielens 1M* to make the size similar to the *Movielens 100K* dataset in the experiments:

1) *MI* (6040×263): This dataset contains ratings for 263 movies randomly selected from 3,952 movies provided by 6,040 users.
2) *MU* (401×3952): This dataset contains ratings for 3,952 movies provided by 401 users randomly selected from 6,040 users.

Likewise, the following two subsets are also extracted from the original *Netflix* dataset to make the size equal to the *Netflix* subset for comparative purpose.

1) *NI* (6800×500): This dataset contains ratings for 500 randomly selected movies provided by 6,800 users.
2) *NU* (955×3561): This dataset contains ratings for 3,561 randomly selected movies provided by 955 users.

Experiments of the proposed *IESVD*, *UESVD* approaches are conducted on the *Movielens 1M* subsets (*MI* and *MU*) and *Netflix* subsets (*NI* and *NU*). The details are shown in Table 5.

TABLE 5: Experimental datasets

Dataset	Size	Number of ratings	Density
MI	6040×263	59005	3.72%
MU	401×3952	70923	4.46%
NI	6800×500	105444	3.10%
NU	955×3561	110818	3.26%

Table 6 to Table 9 show some experimental details of proposed methods on the datasets including the number of selected items and users ($N = N' = 10\%$ for the *Movielens 1M* subsets *MI*, *MU* and $N = N' = 5\%$ for the *Netflix* subsets *NI*, *NU*), the density of sub-matrix, the number of added ratings and the results of different algorithms on the corresponding datasets.

It can be observed that for the datasets of which the number of users is far greater than the number of items (i.e. *MI* and *NI*), *IESVD* could obtain denser sub-matrices. While for datasets with far more items than users (i.e. *MU* and *NU*), the density of the sub-matrices produced by *UESVD* are greater. However, the number of extra ratings predicted from the denser sub-matrix is less than that from the sparser sub-matrices. Therefore, it is inappropriate to compare the results of different algorithms based on the certain number of items and users N. As a result, the best performance (with least *RMSE*) of the proposed algorithms are listed based on the best choices of N (setting $N = N'$).

TABLE 6: Comparison of the proposed methods on *MI* (6040×263)

N=10%	Sub-matrix Density	Extra Ratings	*RMSE* (Best)
RSVD	null	null	1.0432
ESVD	53.80%	7255	1.0286
IESVD	56.58%	6818	1.0235
UESVD	54.64%	7123	1.0246

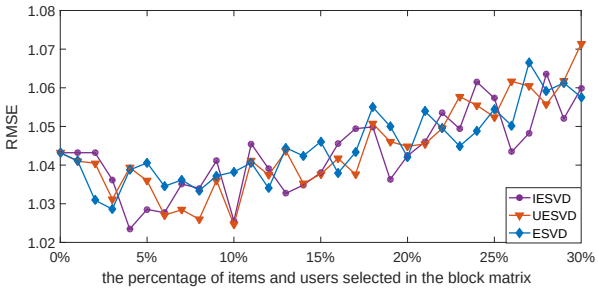


FIGURE 10: *RMSE* of the proposed methods on *MI* (6040×263)

TABLE 7: Comparison of the proposed methods on *MU* (401×3952)

N=10%	Sub-matrix Density	Extra Ratings	*RMSE* (Best)
RSVD	null	null	0.9898
ESVD	57.52%	6712	0.9791
IESVD	58.52%	6554	0.9749
UESVD	58.80%	6509	0.9802

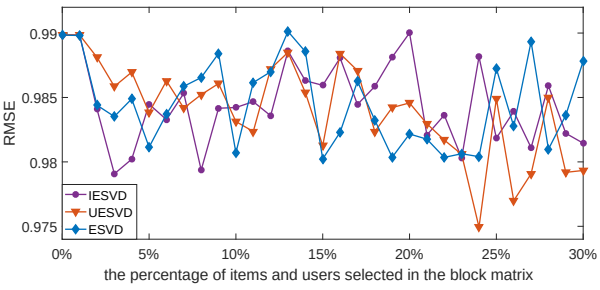


FIGURE 11: *RMSE* of the proposed methods on *MU* (401×3952)

TABLE 8: Comparison of the proposed methods on *NI* (6800×500)

N=5%	Sub-matrix Density	Extra Ratings	*RMSE* (Best)
RSVD	null	null	0.9620
ESVD	59.31%	3459	0.9567
IESVD	66.36%	2859	0.9552
UESVD	61.08%	3308	0.9560

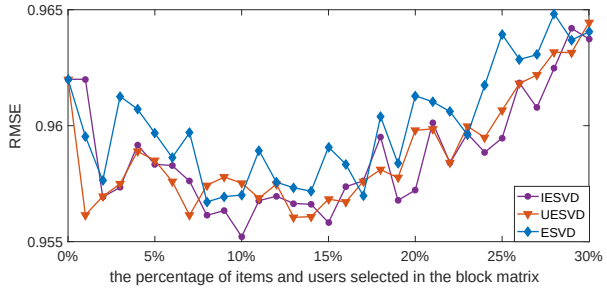


FIGURE 12: *RMSE* of the proposed methods on *NI* (6800×500)

TABLE 9: Comparison of the proposed methods on *NU* (955×3561)

N=5%	Sub-matrix Density	Extra Ratings	*RMSE* (Best)
RSVD	null	null	0.9439
ESVD	62.54%	2038	0.9400
IESVD	68.18%	1717	0.9392
UESVD	68.79%	1684	0.9376

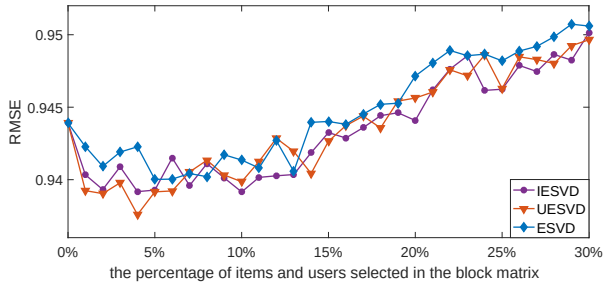


FIGURE 13: *RMSE* of the proposed methods on *NU* (955×3561)

Figure 10 to Figure 13 show the performance (*RMSE*) of the proposed methods with respect to the percentage of items and users (setting $N = N'$) selected in the sub-matrices on *MI*, *MU*, *NI*, *NU* datasets, respectively. It can be seen from the figures all the algorithms start from $N = N' = 0$, where no extra ratings are added into the original matrix. This can be seen as the special case of *RSVD*. As the number of items and users selected in the sub-matrix N goes up, the performance fluctuates. When N is getting large, excessive ratings distort the model and the performance deteriorates. Therefore, the best choices for N that lead to the least *RMSE* are compared. It can be observed that when dealing with the datasets *MI* and *NI* where the number of user is far greater than the number of items, *IESVD* yields a denser sub-matrix than the *UESVD* method. When the datasets contain more items than users (*MU*, *NU*), *UESVD* performs better than *IESVD*.

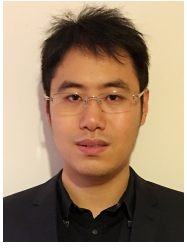
VI. CONCLUSION

The lack of information is an acute challenge in most recommender systems. In this paper, we propose a series of methods, which apply the traditional matrix factorization method that best approximates a given matrix with missing values. Specifically, the general *EVSD* model is firstly proposed by seeking high density through popular items and active users in a special manner inspired by active learning. The corresponding experimental results show its feasibility. The proposed *EVSD* model is further extended into the *MESVD* approach, which learns the model iteratively. The *MESVD* approach can achieve better performance than *ESVD* at the expense of higher computational complexity. In addition, two variants of *ESVD* model are proposed: *IESVD* and *UESVD*. Although the multilayer learning strategy (adopted by *MESVD*) is not applicable to *IESVD* and *UESVD*, their performance are better than *ESVD* when dealing with datasets with imbalanced number of items and users. Instead of viewing active learning from the individual user's point of view, the proposed methods deal with the problem from the system's perspective in a more realistic and effective manner. Although the proposed methods cannot deal with the cold start problem, where the database keeps growing as new users or items continue to be added, it does reduce the computational cost greatly since all the ratings are added simultaneously (*ESVD*, *IESVD* and *UESVD*) or iteratively by a predefined number of iterations (*MESVD*).

REFERENCES

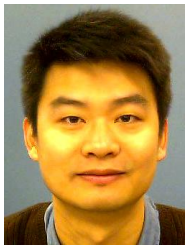
[1] A. S. Das, M. Datar, A. Garg, and S. Rajaram, "Google news personalization: scalable online collaborative filtering," in Proceedings of the 16th international conference on World Wide Web. ACM, 2007, pp. 271–280.
[2] E.-A. Baatarjav, S. Phithakkitnukoon, and R. Dantu, "Group recommendation system for facebook," in On the Move to Meaningful Internet Systems: OTM 2008 Workshops. Springer, 2008, pp. 211–219.
[3] G. Linden, B. Smith, and J. York, "Amazon.com recommendations: Item-to-item collaborative filtering," vol. 7, no. 1. IEEE, 2003, pp. 76–80.
[4] J. Bennett, S. Lanning et al., "The netflix prize," in Proceedings of KDD cup and workshop, vol. 2007. New York, NY, USA, 2007, p. 35.
[5] J. Schafer, D. Frankowski, J. Herlocker, and S. Sen, "Collaborative filtering recommender systems," The adaptive web, pp. 291–324, 2007.
[6] P. Lops, M. De Gemmis, and G. Semeraro, "Content-based recommender systems: State of the art and trends," in Recommender systems handbook. Springer, 2011, pp. 73–105.
[7] S. Trewin, "Knowledge-based recommender systems," Encyclopedia of library and information science, vol. 69, no. Supplement 32, p. 180, 2000.
[8] R. Burke, "Hybrid recommender systems: Survey and experiments," User modeling and user-adapted interaction, vol. 12, no. 4, pp. 331–370, 2002.
[9] G. Adomavicius and A. Tuzhilin, "Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions," IEEE transactions on knowledge and data engineering, vol. 17, no. 6, pp. 734–749, 2005.
[10] J. S. Breese, D. Heckerman, and C. Kadie, "Empirical analysis of predictive algorithms for collaborative filtering," in Proceedings of the Fourteenth conference on Uncertainty in artificial intelligence. Morgan Kaufmann Publishers Inc., 1998, pp. 43–52.
[11] B. Sarwar, G. Karypis, J. Konstan, and J. Riedl, "Item-based collaborative filtering recommendation algorithms," in Proceedings of the 10th international conference on World Wide Web. ACM, 2001, pp. 285–295.
[12] R. M. Bell and Y. Koren, "Scalable collaborative filtering with jointly derived neighborhood interpolation weights," in Data Mining, Seventh IEEE International Conference on. IEEE, 2007, pp. 43–52.
[13] T. Hofmann, "Latent semantic models for collaborative filtering," ACM Transactions on Information Systems (TOIS), vol. 22, no. 1, pp. 89–115, 2004.
[14] R. Salakhutdinov and A. Mnih, "Probabilistic matrix factorization." in Nips, vol. 1, no. 1, 2007, pp. 2–1.
[15] S. Rendle, "Factorization machines with libfm," ACM Transactions on Intelligent Systems and Technology (TIST), vol. 3, no. 3, p. 57, 2012.
[16] Y. Koren, "Factorization meets the neighborhood: a multifaceted collaborative filtering model," in Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining. ACM, 2008, pp. 426–434.

[17] A. Paterek, "Improving regularized singular value decomposition for collaborative filtering," in Proceedings of KDD cup and workshop, vol. 2007, 2007, pp. 5–8.
[18] X. Guan, C.-T. Li, and Y. Guan, "Enhanced svd for collaborative filtering," in Pacific-Asia Conference on Knowledge Discovery and Data Mining. Springer, 2016, pp. 503–514.
[19] Y. Koren, R. Bell, and C. Volinsky, "Matrix factorization techniques for recommender systems," Computer, no. 8, pp. 30–37, 2009.
[20] X. Su and T. M. Khoshgoftaar, "A survey of collaborative filtering techniques," Advances in artificial intelligence, vol. 2009, p. 4, 2009.
[21] N. Rubens, M. Elahi, M. Sugiyama, and D. Kaplan, "Active learning in recommender systems," in Recommender systems handbook. Springer, 2015, pp. 809–846.
[22] M. Elahi, F. Ricci, and N. Rubens, "A survey of active learning in collaborative filtering recommender systems," Computer Science Review, vol. 20, pp. 29–50, 2016.
[23] A. S. Harpale and Y. Yang, "Personalized active learning for collaborative filtering," in Proceedings of the 31st annual international ACM SIGIR conference on Research and development in information retrieval. ACM, 2008, pp. 91–98.
[24] R. Jin and L. Si, "A bayesian approach toward active learning for collaborative filtering," in Proceedings of the 20th conference on Uncertainty in artificial intelligence. AUAI Press, 2004, pp. 278–285.
[25] M. Elahi, F. Ricci, and N. Rubens, "Active learning strategies for rating elicitation in collaborative filtering: a system-wide perspective," ACM Transactions on Intelligent Systems and Technology (TIST), vol. 5, no. 1, p. 13, 2013.
[26] B. N. Miller, I. Albert, S. K. Lam, J. A. Konstan, and J. Riedl, "Movielens unplugged: experiences with an occasionally connected recommender system," in Proceedings of the 8th international conference on Intelligent user interfaces, 2003, pp. 263–266.
[27] B. Sarwar, G. Karypis, J. Konstan, and J. Riedl, "Analysis of recommendation algorithms for e-commerce," in Proceedings of the 2nd ACM conference on Electronic commerce. ACM, 2000, pp. 158–167.
[28] S. Funk, "Netflix update: Try this at home," 2006.
[29] R. Gemulla, E. Nijkamp, P. J. Haas, and Y. Sismanis, "Large-scale matrix factorization with distributed stochastic gradient descent," in Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining. ACM, 2011, pp. 69–77.
[30] A. M. Rashid, I. Albert, D. Cosley, S. K. Lam, S. M. McNee, J. A. Konstan, and J. Riedl, "Getting to know you: learning new user preferences in recommender systems," in Proceedings of the 7th international conference on Intelligent user interfaces. ACM, 2002, pp. 127–134.
[31] N. Golbandi, Y. Koren, and R. Lempel, "On bootstrapping recommender systems," in Proceedings of the 19th ACM international conference on Information and knowledge management. ACM, 2010, pp. 1805–1808.
[32] B. M. Marlin, R. S. Zemel, S. T. Roweis, and M. Slaney, "Recommender systems, missing data and statistical model estimation." in IJCAI, 2011, pp. 2686–2691.
[33] G. Carenini, J. Smith, and D. Poole, "Towards more conversational and collaborative recommender systems," in Proceedings of the 8th international conference on Intelligent user interfaces. ACM, 2003, pp. 12–18.



Xin Guan received the B.S. degree in information and computing science from the China University of Mining and Technology, Beijing, China, in 2011, and the M.S. degree in financial mathematics and computation from the University of Leicester, Leicester, UK, in 2012. He is currently working toward the Ph.D. degree with the Department of Computer Science, University of Warwick, Coventry, UK. His research interests include machine learning, data mining and recommendation systems.

Chang-Tsun Li received the BEng degree in electrical engineering from National Defence University (NDU), Taiwan, in 1987, the MSc degree in computer science from U.S. Naval Postgraduate School, USA, in 1992, and the PhD degree in computer science from the University of Warwick, UK, in 1998. He was an associate professor of the Department of Electrical Engineering at NDU during 1998-2002 and a visiting professor of the Department of Computer Science at U.S. Naval Postgraduate School in the second half of 2001. He was a professor of the Department of Computer Science at the University of Warwick, UK, until Dec 2016. He is currently a professor of the School of Computing and Mathematics, Charles Sturt University, Australia, leading the Data Science Research Unit. His research interests include multimedia forensics and security, biometrics, data mining, machine learning, data analytics, computer vision, image processing, pattern recognition, bioinformatics, and content-based image retrieval. The outcomes of his multimedia forensics and machine learning research have been translated into award-winning commercial products protected by a series of international patents and have been used by a number of police forces and courts of law around the world. He is currently Associate Editor of the EURASIP Journal of Image and Video Processing (JIVP) and Associate of Editor of IET Biometrics. He involved in the organisation of many international conferences and workshops and also served as member of the international program committees for several international conferences. He is also actively contributing keynote speeches and talks at various international events.



Yu Guan received his PhD degree at the Department of Computer Science, University of Warwick in 2015. Then he moved to the School of Computing Science, Newcastle University for his postdoc research, before becoming a lecturer there in 2017. His research interests are machine learning, deep learning, ubiquitous computing, and computer vision. He has published several papers at the top venues including IEEE Trans. Pattern Analysis and Machine Intelligence (T-PAMI), ACM Interactive, Mobile, Wearable and Ubiquitous Technologies (IMWUT), etc. He is currently the Associate Editor of ACM IMWUT, and a regular reviewer for several venues such as IEEE T-PAMI, IEEE T-IFS, IEEE T-CSVT, IEEE T-CYB, IEEE ISWC, ACM MM, ACM UbiComp, PRL, SPL, IVC, etc.